CINDI: COARSE-TO-FINE IMPLICIT NEURAL DIFFUSION FOR EFFICIENT PIXEL-SPACE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pixel-space diffusion and flow-based models typically perform every denoising step at the full image resolution. Although early steps primarily establish coarse structure, sampling them still incurs the full cost of fine-resolution processing. We ask whether spatial resolution can instead be scheduled along the sampling trajectory, allowing early function evaluations to operate on coarser grids while preserving final generation quality. To answer this question, we present **CINDI** (**C**oarse-to-fine **I**mplicit **N**eural **DI**ffusion), a pixel-space generative model that progressively increases spatial resolution along the sampling trajectory. CINDI features an end-to-end implicit transformer denoiser trained with the next-scale bridge loss to enable both in-scale and cross-scale prediction, and a companion coarse-to-fine sampler that operates without the need for multi-model cascading or latent-space compression. We compare CINDI against a single-scale JiT baseline that achieves state-of-the-art-level performance on ImageNet. With matched model size, training epochs, and sampling steps, CINDI achieves competitive FID scores while delivering up to a 3× speedup in inference throughput. Additionally, benefiting from its multiscale prediction capability, CINDI also natively supports generation beyond its training resolution without extra fine-tuning.

1 INTRODUCTION

Modern diffusion and flow-based image generators achieve strong sample quality (Ho et al., 2020; Song et al., 2021; Lipman et al., 2023) by iteratively transforming noise into images. Transformer denoisers such as DiT (Peebles & Xie, 2023) and JiT (Li & He, 2026) scale this recipe further with powerful sequence models, but their sampling cost grows rapidly as the image resolution increases. A major inefficiency comes from enduring full-resolution cost at every denoising step, even though early steps primarily establish coarse semantic structure and global layout.

Existing efficiency strategies tackle this problem from different perspectives, but leave important gaps. Latent diffusion reduces spatial cost by moving generation into a compressed representation, but the final image quality remains constrained by the pretrained encoder-decoder (Rombach et al., 2022). Cascaded diffusion embraces the coarse-to-fine paradigm, but typically requires multiple models stacking instead of a shared end-to-end solution (Ho et al., 2022; Saharia et al., 2022). Implicit neural representations (INRs) (Sitzmann et al., 2020; Kim & Kim, 2024) offer the potential for continuous image synthesis through coordinate-based queries; however, most existing INR-based generative approaches leverage this property only after generation, treating it as a post-hoc rendering step rather than integrating it into the generative process itself for efficient sampling (Sitzmann et al., 2020; Kim & Kim, 2024).

This observation motivates our central question: Can we schedule spatial resolution along a diffusion trajectory by leveraging the continuous synthesis capabilities of an INR, instead of relying on latent compression or model stacking?

To answer this question, we present CINDI (Coarse-to-fine Implicit Neural Diffusion), a pixel-space generative model that integrates an INR directly into the denoising process for cross-scale prediction, rather than using it merely as a post-hoc decoder. CINDI employs a shared denoiser with a two-resolution interface: a Transformer encodes the noisy state at the active resolution, while a coordinate-based INR learns a continuous function that predicts the clean image at either the active or the next finer resolution.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

During training, CINDI supervises both in-scale denoising and next-scale prediction, so the implicit denoiser learns to transfer a coarse noisy state into a finer clean target. During sampling, the model performs short ordinary differential equation (ODE) segments on a scale ladder, predicts the next scale at each boundary, then adds back noise to the rendered prediction to continue the trajectory.

We evaluate CINDI on class-conditional ImageNet (Deng et al., 2009) at 64×64 and 256×256 against JiT (Li & He, 2026), a single-scale Diffusion Transformer baseline with state-of-the-art (SOTA) level performance. Our findings shows that under matched model size, training epochs, and sampling steps, moving early ODE steps onto cheaper coarse grids can substantially speed up inference throughput while maintaining competitive generation quality. With matched 200 training epochs and 50 sampling steps, CINDI's best-performing two-scale schedule edges out JiT in Fréchet Inception Distance (FID) (Heusel et al., 2017) while achieving up to a $3\times$ speedup in inference throughput at varies matched resolution and model scale.

Our contributions are:

1. CINDI, a unified coarse-to-fine implicit neural diffusion model that enables progressive spatial resolution scheduling through learned INR functions, without requiring latent compression or model stacking.
2. An implicit denoising interface that decouples the clean-prediction resolution from the active noisy-state resolution, supporting both in-scale and cross-scale predictions.
3. A grouped coarse-to-fine training objective with an explicit next-scale bridge loss that provides a second anchor to guide cross-scale progression and significantly improves structural preservation.
4. A comprehensive evaluation on ImageNet demonstrating that CINDI is flexible and efficient across different target resolutions and scale ladder configurations, and our best-performing configuration edges out the SOTA-level JiT baseline in FID while delivering up to a $3\times$ speedup in inference throughput at matched model size, training epochs, and sampling steps.

2 Related Work

2.1 Pixel-Space and Latent-Space Diffusion

Diffusion models can be broadly categorized by the space in which denoising is performed. Pixel-space diffusion directly models image signals without an intermediate representation bottleneck, but each denoising evaluation operates on the full-resolution image grid, making high-resolution generation computationally expensive (Ho et al., 2020; Song et al., 2021). Latent diffusion models, in contrast, first encode images into lower-dimensional representations and perform denoising in the compressed latent space, substantially reducing the computational cost of high-resolution synthesis (Rombach et al., 2022). However, their final output quality is inherently constrained by the reconstruction fidelity of the pretrained encoder-decoder. CINDI follows the pixel-space route to preserve direct signal modeling while improving sampling efficiency by reducing the amount of full-grid computation required during generation.

2.2 Implicit neural representations

Implicit Neural Representations model signals as continuous neural functions rather than discrete grids, mapping spatial coordinates directly to their corresponding signal values (Sitzmann et al., 2020). This coordinate-based formulation has been widely adopted for images, videos, 3D shapes, and neural rendering (Sitzmann et al., 2020; Park et al., 2019; Mescheder et al., 2019; Mildenhall et al., 2020). By representing signals continuously, INRs enable flexible querying at arbitrary coordinates, compact signal representation, and resolution-independent reconstruction. These properties make INRs particularly suitable for generative modeling that aims to produce samples across multiple scales or beyond the original target resolution.

However, many INR-based generative approaches use INRs mainly as a representation space or final decoder. A common pattern is to train a latent diffusion model together with a separate implicit encoder–decoder pair (Kim & Kim, 2024; Chen et al., 2024). This enables resolution-independent querying, but the capability is usually exploited only after the generative trajectory has been completed, leaving its potential for cross-scale generation largely unused.

2.3 Diffusion, Flow Matching, and Transformer Denoisers

Diffusion models learn generative dynamics that transport noise into data through denoising or score estimation along a prescribed probability path (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021). Flow matching provides a compatible training formulation that directly regresses the velocity field of this transport (Liu et al., 2022; Lipman et al., 2023), while stochastic interpolants place both views within a shared path-based framework (Albergo et al., 2023). These path-based formulations are increasingly paired with scalable Transformer denoisers in place of convolutional U-Nets (Ronneberger et al., 2015). Diffusion Transformers (DiTs) establish this direction in latent space, showing consistent gains with model capacity and training compute (Vaswani et al., 2017; Dosovitskiy et al., 2021; Peebles & Xie, 2023), while Scalable Interpolant Transformers (SiTs) retain the DiT backbone under a flexible interpolant framework that separates the probability path, training objective, and sampling process (Ma et al., 2024). JiT brings these directions together in pixel space, combining a DiT-style large-patch Transformer with a linear flow-matching formulation and clean-image prediction (Li & He, 2026). CINDI follows this self-contained pixel-space transformer direction, but decouples the clean-image prediction grid from the active noisy grid through a learned continuous function, enabling progressive generation via coarse-to-fine implicit rendering.

2.4 Cascaded, Multiscale, and Coarse-to-Fine Generation

Cascaded diffusion models decompose high-resolution generation into multiple stages. A low-resolution base model first generates a coarse sample, which is then progressively refined by one or more super-resolution diffusion models (Ho et al., 2022; Saharia et al., 2022). Although this design reduces the difficulty of each stage, it requires training and maintaining multiple models. Each handoff also forms a boundary between separately optimized components, where later stages must recover fine structure from a fixed coarse prediction rather than a shared generative state.

Recent work has explored unified multiscale generation without separate model cascades. Matryoshka Diffusion Models (MDM) jointly model multiple pixel-space resolutions with a NestedUNet, but denoise all resolution states in parallel rather than assigning different trajectory segments to progressively finer grids (Gu et al., 2023). Visual Autoregressive Modeling (VAR) performs coarse-to-fine next-scale prediction, but relies on a separately trained multiscale VQ-VAE and operates over discrete latent tokens (Tian et al., 2024). PixelFlow similarly employs a shared pixel-space flow model for coarse-to-fine generation (Chen et al., 2025), but relies on predefined upsampling operations at scale transitions and requires fine-grained patch representations to maintain quality.

CINDI embraces the same broad coarse-to-fine intuition and targets efficient pixel-space image generation. Its shared implicit denoiser learns both in-scale denoising and next-scale bridge prediction, allowing scale transitions to be trained as part of a single end-to-end generative process without model cascades or latent compression.

3 Preliminaries

CINDI is a pixel space DiT variant, specifically adopting the JiT formulation. We begin by describing the common formulation and parameterization in this section.

3.1 Linear Interpolation Path

We use the linear noise-to-data interpolation commonly adopted in rectified flow, flow matching, and stochastic interpolants (Liu et al., 2022; Lipman et al., 2023; Albergo et al., 2023):

$$\mathbf{z}_t = t\,\mathbf{x} + (1-t)\,\boldsymbol{\epsilon}, \qquad \mathbf{x} \sim p_{\text{data}}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I), \quad t \in [0,1]. \tag{1}$$

For a sampled pair $(\mathbf{x}, \boldsymbol{\epsilon})$, differentiating the path gives the sample-wise target velocity

$$v = \mathbf{x} - \boldsymbol{\epsilon} = \frac{\mathbf{x} - \mathbf{z}_t}{1-t}, \qquad t \in [0,1). \tag{2}$$

The model learns the corresponding velocity field $v_\theta(\mathbf{z}_t, t)$. Generation then integrates the associated ODE from the noise distribution at $t = 0$ to the data distribution at $t = 1$:

$$\frac{\mathrm{d}\mathbf{z}_t}{\mathrm{d}t} = v_\theta(\mathbf{z}_t, t). \tag{3}$$

3.2 DENOISING PARAMETERIZATION

We parameterize the denoiser to predict the clean image $\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t)$ rather than noise or velocity. Clean-image prediction combined with a velocity-space loss consistently outperforms noise- and velocity-prediction alternatives under the same loss formulation (Li & He, 2026). We obtain the corresponding velocity field by substituting the clean prediction for $\mathbf{x}$ in Eq. (2):

$$v_\theta(\mathbf{z}_t, t) = \frac{\hat{\mathbf{x}}_\theta(\mathbf{z}_t, t) - \mathbf{z}_t}{\max(1-t,\, t_\epsilon)}, \qquad t_\epsilon > 0. \tag{4}$$

The target is computed with the same denominator, $v_{\text{target}} = (\mathbf{x} - \mathbf{z}_t)/\max(1-t, t_\epsilon)$. The velocity-regression objective is

$$\mathcal{L}_{\text{vel}} = \mathbb{E}_{\mathbf{x},\,\boldsymbol{\epsilon},\,t} \left\| v_\theta(\mathbf{z}_t, t) - v_{\text{target}} \right\|_2^2, \tag{5}$$

3.3 ODE SAMPLING

Sampling integrates the learned velocity field from $\mathbf{z}_0 \sim \mathcal{N}(0, \sigma^2 I)$ over $0 = t_0 < t_1 < \ldots < t_K = 1$. Writing $\mathbf{z}_{t_k}$ for the state at t_k and $\Delta t_k = t_{k+1} - t_k$, the Euler and Heun updates are

$$\begin{aligned} \mathbf{z}' &= \mathbf{z}_{t_k} + \Delta t_k\, v_\theta(\mathbf{z}_{t_k}, t_k), \\ \mathbf{z}^{\text{Euler}}_{t_{k+1}} &= \mathbf{z}', \\ \mathbf{z}^{\text{Heun}}_{t_{k+1}} &= \mathbf{z}_{t_k} + \frac{\Delta t_k}{2}\left[v_\theta(\mathbf{z}_{t_k}, t_k) + v_\theta(\mathbf{z}', t_{k+1}) \right]. \end{aligned} \tag{6}$$

In our sampler, we always use Euler for boundary steps, so a K-step trajectory requires K velocity evaluations with Euler and $2K - 1$ with Heun (Karras et al., 2022) .

4 METHOD: CINDI

Let $\mathcal{R} = \{r_0, r_1, \ldots, r_{S-1}\}$ be an ordered scale ladder with $r_0 < r_1 < \cdots < r_{S-1}$, where $r_{S-1} = H = W$ is the highest target resolution. The index $s \in \{0, \ldots, S-1\}$ denotes the active scale and r_s its spatial resolution. With patch size p, the Transformer processes $N_s = (r_s/p)^2$ image tokens, so coarser active scales reduce its per-evaluation cost.

CINDI parameterizes a clean-image predictor $\hat{\mathbf{x}}_\theta(\mathbf{z}_t^{r_s}, t, y;\, r_s, r_{\text{out}})$ with two resolution arguments. Here, y is the class label. The active resolution r_s determines the noisy state grid, whereas r_{out} determines the clean prediction grid. Setting $r_{\text{out}} = r_s$ gives in-scale denoising; setting $r_{\text{out}} = r_{s+1}$ gives the next-scale bridge prediction.

As shown in Figure 1, the denoiser combines a scale-adaptive Transformer encoder E_θ with a coordinate-based implicit renderer R_θ. Both are conditioned on the same vector c_s:

$$c_s = \text{emb}_t(t) + \text{emb}_y(y) + \text{emb}_s(s). \tag{7}$$

The learned scale embedding distinguishes segments that share the same flow time but use different active resolutions or noise scales.

4.1 SCALE-ADAPTIVE IMPLICIT DENOISER

The encoder part of the denoiser retains the JiT Transformer backbone (Li & He, 2026), but shares one set of weights across variable active grids. Its patch-token count and positional representation adapt to r_s. It maps the active noisy state to a contextualized N_s-token latent field:

$$\mathbf{h}_s = E_\theta(\mathbf{z}_t^{r_s}, c_s;\, r_s). \tag{8}$$

The implicit renderer R_θ queries a continuous image function from $\mathbf{h}_s$. Each output pixel has global and active-cell local coordinates $\mathbf{u} = (x_g, y_g, x_\ell, y_\ell) \in [-1, 1]^4$. The routing function $j_{s,r_{\text{out}}}(\mathbf{u}) \in \{1, \ldots, N_s\}$ selects its active token. The adapter A_θ produces layer-wise hierarchical token features $\{\mathbf{m}_s^q\}_q$, which condition the coordinate MLP:

$$\begin{aligned} \{\mathbf{m}_s^q\}_q &= A_\theta(\mathbf{h}_s, c_s), \\ \hat{\mathbf{p}}_\theta(\mathbf{u}) &= f_\theta(\phi(\mathbf{u}); \{\mathbf{m}_s^q[j_{s,r_{\text{out}}}(\mathbf{u})]\}_q) \in \mathbb{R}^3. \end{aligned} \tag{9}$$

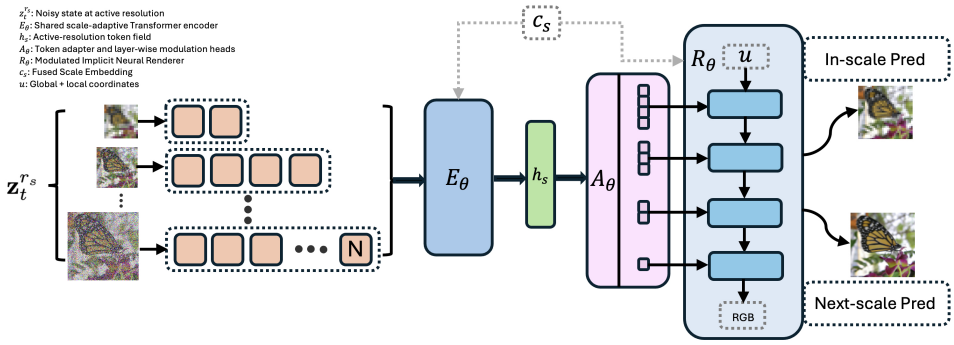

Figure 1: CINDI's two-resolution implicit denoiser. The scale-adaptive Transformer E_θ encodes the active noisy state $\mathbf{z}_t^{r_s}$ into token field $\mathbf{h}_s$. The adapter and layer-wise modulation heads A_θ condition the implicit renderer R_θ, which is queried using global and local coordinates $\mathbf{u}$. The same latent field supports in-scale prediction when $r_{\text{out}} = r_s$ and next-scale bridge prediction when $r_{\text{out}} = r_{s+1}$.

Here, ϕ is a Fourier coordinate encoding (Tancik et al., 2020), q indexes the layer-wise modulation heads, and f_θ applies their routed features through feature-wise linear modulation (FiLM) (Perez et al., 2018). Evaluating all coordinates on the requested grid produces R_θ's image output. When $r_{\text{out}} > r_s$, distinct coordinates allow multiple output pixels to share one active token, enabling the same latent field to support in-scale, next-scale, and beyond-training-scale prediction.

Combining both components gives the complete denoiser:

$$\hat{\mathbf{x}}_\theta(\mathbf{z}_t^{r_s}, t, y;\, r_s, r_{\text{out}}) = R_\theta\big(E_\theta(\mathbf{z}_t^{r_s}, c_s;\, r_s),\, c_s;\, r_s, r_{\text{out}}\big). \tag{10}$$

4.2 Multiscale Training and Bridge Supervision

At each training iteration, CINDI partitions a minibatch of B examples across the scale ladder so that all resolutions are optimized jointly with shared model weights. Let ρ_s denote the ratio of examples allocated to scale s, where $\sum_{s=0}^{S-1} \rho_s = 1$. Thus, scale s receives approximately $\rho_s B$ examples from each minibatch.

For an example assigned to scale s, we draw a normalized segment time $\tau \in (0, 1)$ and map it to the scale-specific interval $[a_s, b_s] \subseteq [0, 1]$. The inverse relation is $\tau = (t_s - a_s)/(b_s - a_s)$. The clean image is resized to $\mathbf{x}^{r_s}$ and corrupted using noise magnitude σ_s, obtained from the per-scale coefficient α_s. We denote the resulting active state by $\mathbf{z}_s$:

$$\begin{aligned} t_s &= a_s + (b_s - a_s)\tau, \\ \sigma_s &= \alpha_s(r_s/256), \\ \boldsymbol{\epsilon}_s &\sim \mathcal{N}(0, \sigma_s^2 I), \\ \mathbf{z}_s &= t_s \mathbf{x}^{r_s} + (1 - t_s)\boldsymbol{\epsilon}_s. \end{aligned} \tag{11}$$

Per-scale noise coefficient. At image resolution r, JiT uses a fixed coefficient to set $\sigma_{\text{JiT}} = 1.0 \times (r/256)$ and samples $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_{\text{JiT}}^2 I)$ (Li & He, 2026). CINDI instead allows coefficient α_s to vary by scale. The learned active-scale embedding in Eq. (7) distinguishes tokens from different time and noise-level combinations.

In-scale denoising loss. At the active resolution, CINDI predicts $\hat{\mathbf{x}}_{s\to s} = \hat{\mathbf{x}}_\theta(\mathbf{z}_s, t_s, y;\, r_s, r_s)$. Following Eq. (5), its velocity loss and equivalent weighted clean-image form are

$$\ell_{\text{main}}^s = \left\| \frac{\hat{\mathbf{x}}_{s\to s} - \mathbf{z}_s}{\max(1 - t_s, t_\epsilon)} - \frac{\mathbf{x}^{r_s} - \mathbf{z}_s}{\max(1 - t_s, t_\epsilon)} \right\|_2^2 = \left\| \frac{\hat{\mathbf{x}}_{s\to s} - \mathbf{x}^{r_s}}{\max(1 - t_s, t_\epsilon)} \right\|_2^2. \tag{12}$$

Next-scale bridge loss. For each non-final scale, the same active noisy state $\mathbf{z}_s$ is retained as input, but the renderer is queried at r_{s+1}:

$$\hat{\mathbf{x}}_{s\to s+1} = \hat{\mathbf{x}}_\theta(\mathbf{z}_s, t_s, y;\, r_s, r_{s+1}). \tag{13}$$

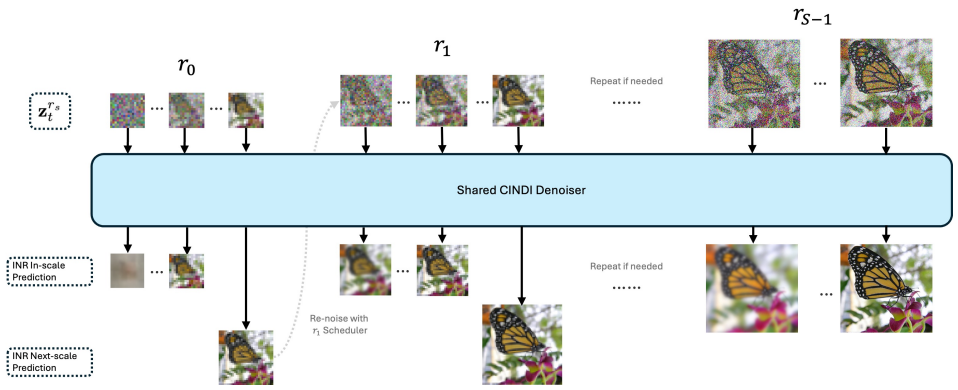

Figure 2: Coarse-to-fine sampling in CINDI. The shared denoiser evolves noisy states through local ODE segments at resolutions $r_0, r_1, \ldots, r_{S-1}$, producing in-scale clean predictions within each segment. At every non-final boundary, the INR renders a clean prediction at the next resolution, which has noise added back to it using the destination scale's schedule to initialize the following segment. The process repeats until the final-resolution image is generated.

Using the same weighted clean-image form and the target $\mathbf{x}^{r_{s+1}}$, obtained by resizing the original image directly to the next resolution, we define

$$\ell^s_{\text{next}} = \left\| \frac{\hat{\mathbf{x}}_{s\to s+1} - \mathbf{x}^{r_{s+1}}}{\max(1 - t_s, t_\epsilon)} \right\|_2^2, \qquad s < S - 1. \tag{14}$$

With in-scale supervision alone, the INRs is constrained with only one anchor in its cross-scale behavior, so its prediction on a finer grid remains random and underdetermined. Pairing in-scale and next-scale targets gives the same latent state two resolution anchors, defining the desired coarse-to-fine direction and guide the renderer to better preserve the image structure at crossing.

With the next-scale term omitted for $s = S - 1$, the per-scale and overall objectives are

$$\ell^s = \ell^s_{\text{main}} + \lambda_{\text{next}} \ell^s_{\text{next}}, \qquad \mathcal{L} = \sum_{s=0}^{S-1} \rho_s \, \mathbb{E}[\ell^s]. \tag{15}$$

4.3 Cross-scale sampling

At inference, CINDI replaces one full-resolution ODE trajectory with a sequence of local trajectories over $\mathcal{R}$. Let K_s denote the number of ODE steps assigned to scale segment s, extending the single-scale count K from the preliminaries. Segment s integrates $\mathbf{z}_t^{r_s}$ over $t \in [a_s, b_s]$ using K_s uniformly spaced Euler or Heun updates and Eq. (10) with $r_{\text{out}} = r_s$. Sampling begins from $\mathbf{z}_{a_0}^{r_0} \sim \mathcal{N}(0, \sigma_0^2 I)$ at the coarsest resolution, where $a_0 = 0$, and the final segment satisfies $b_{S-1} = 1$.

At the endpoint of each non-final segment, CINDI renders the next-scale clean prediction and re-noises it using the destination scale's time and noise settings:

$$\begin{aligned} \hat{\mathbf{x}}_{s\to s+1} &= \hat{\mathbf{x}}_\theta(\mathbf{z}_{b_s}^{r_s}, b_s, y;\, r_s, r_{s+1}), \qquad s < S - 1, \\ \mathbf{z}_{a_{s+1}}^{r_{s+1}} &= a_{s+1} \hat{\mathbf{x}}_{s\to s+1} + (1 - a_{s+1}) \boldsymbol{\epsilon}_{s+1}, \quad \boldsymbol{\epsilon}_{s+1} \sim \mathcal{N}(0, \sigma_{s+1}^2 I). \end{aligned} \tag{16}$$

This transition repeats until the full-resolution segment is reached. See Figure 2 for an overview.

The total sampling-step budget is $K_{\text{total}} = \sum_s K_s$. With the Heun sampler used in our experiments, the per-segment final Euler steps and the $S-1$ bridge predictions yield $2K_{\text{total}} - 1$ model predictions, matching a single-scale K_{total}-step Heun trajectory. Users can flexibly adjust number of segments during training and the number of steps per segment during sampling.

5 Experiments

We evaluate around four questions. First, at matched model scale, training epochs, and sampling budget, can CINDI preserve JiT's generation quality while dramatically improving inference

throughput? Second, what is the effect of explicit next-scale loss on cross-scale behavior? Third, how do the starting resolution, scale ladder, and the step allocation control the quality–efficiency trade-off? Finally, how does CINDI perform when compared against other modern generative models?

5.1 Experimental protocol

Datasets and models. We evaluate class-conditional generation on ImageNet (Deng et al., 2009). ImageNet 64×64 is used for controlled studies and finding the best recipe. The preferred configuration is then transferred to ImageNet 256×256 to test CINDI's high-resolution capability. We compare CINDI-B/4 with JiT-B/4 at 64×64 and CINDI-L/H/16 with JiT-L/H/16 at 256×256.

We also compare CINDI's 256 variants with other modern generative models. However since each external models have different training budgets, sampling steps, and precision settings, we discuss only a system-level comparison rather than an algorithmically matched benchmark. Our direct speedup claims remain restricted to the matched CINDI–JiT comparisons.

Metrics and fairness. We use FID to evaluate generation quality and measure inference throughput in images per second to assess efficiency. For direct comparisons between CINDI and JiT, we match model scale, training duration (200 epochs), total sampling steps ($K_{\text{total}} = 50$), number of function evaluations (99 without CFG and 198 with CFG), sampling method (Heun), and runtime settings. Both models are evaluated using interval classifier-free guidance (CFG). For throughput, we use 1xH200 GPU with batch size 256 for repeated measurements before reporting the average. Full experimental settings are provided in Appendix A.

We note that the published JiT-B/4 result does not incorporate the general-purpose backbone improvements used in the other JiT-B/L/H variants and in our CINDI variants. For a fair comparison, we therefore retrain JiT-B/4 with the same backbone improvements (RoPE, qk-norm, and in-context class tokens (Su et al., 2024; Henry et al., 2020; Dosovitskiy et al., 2021; Li et al., 2024)) and report both the original and retrained results.

For comparison with other modern generative models, we conduct same throughput measurements on selected models where pretrained checkpoints are available. Settings are provided in Appendix C.

5.2 Results on ImageNet 64

We begin on ImageNet 64×64, where controlled experiments isolate how the starting resolution, intermediate scale ladder, and next-scale supervision shape CINDI's quality–efficiency trade-off. Table 1 reports these variants alongside the matched JiT-B/4 baseline.

Table 1: ImageNet 64×64 CINDI vs JiT comparison and next-scale supervision ablation. For JiT-B/4, official number is shown first, our Section 5.1 reproduction is shown in parentheses.

Variant	Scale ladder	λ_{next}	Params	FID-50k ↓	FP32 img/s ↑	FP32 speedup	BF16 img/s ↑	BF16 speedup
JiT-B/4 (Li & He, 2026)	64	–	131M	3.55 (2.21)	4.16	1.00×	18.08	1.00×
CINDI-B/4	32→64	0.1	133M	**2.19**	8.05	1.93×	54.62	3.02×
CINDI-B/4	16→64	0.1	133M	3.96	**10.56**	**2.53×**	**67.55**	**3.73×**
CINDI-B/4	16→32→64	0.1	133M	3.47	10.05	2.41×	64.96	3.59×
CINDI-B/4	16→32→48→64	0.1	133M	3.21	8.83	2.12×	57.17	3.16×
CINDI-B/4 w/o next-scale loss	32→64	0.0	133M	2.75	8.05	1.93×	54.62	3.02×
CINDI-B/4 w/o next-scale loss	16→64	0.0	133M	4.35	10.56	2.53×	67.55	3.73×

Matched quality with higher throughput. The best CINDI-B/4 configuration, $32 \rightarrow 64$, reaches FID 2.19. This is numerically better than both the 3.55 reported for JiT-B/4 and our stronger 2.21 reproduction with the same general-purpose components. At comparable generation quality, CINDI substantially improves inference efficiency, reaching 8.05 versus 4.16 images/s in FP32 ($1.93\times$) and 54.62 versus 18.08 images/s in BF16 ($3.02\times$). This advantage is consistent across numerical precisions and becomes more pronounced under BF16.

Starting-resolution trade-off. The starting resolution strongly controls the quality–efficiency balance. Moving the initial grid from 32 to 16 increases throughput from 8.05 to 10.56 images/s in FP32



Figure 3: Effect of next-scale supervision on a $16 \rightarrow 64$ transition. Top: without the next-scale loss, next-scale prediction is more pixelated with weak structure. Bottom: with the next-scale loss, jump prediction better preserves the silhouette and wing geometry, leading to a more natural final image.

and from 54.62 to 67.55 images/s in BF16, but worsens FID from 2.19 to 3.96. Thus, assigning more computation to a very coarse grid can make sampling faster, but a single large $16 \rightarrow 64$ transition discards too much spatial structure to preserve the best generation quality.

Progressive scale ladders. Intermediate scales could recover much of the quality lost from the aggressive 16-pixel starting resolution. Relative to direct $16 \rightarrow 64$ sampling, the $16 \rightarrow 32 \rightarrow 64$ ladder improves FID from 3.96 to 3.47, while introducing an additional 48-pixel stage further reduces it to 3.21. Across the tested schedules, smaller jumps between adjacent resolutions consistently lead to better generation quality. This improvement comes at the cost of additional computation on finer, more expensive grids , reducing BF16 throughput from 67.55 to 64.96 and 57.17 images/s, respectively. Compare to JiT-B/4, the four-scale ladder remains $2.12\times$ faster in FP32 and $3.16\times$ faster in BF16.

Next-scale loss is essential. Next-scale loss is the key to CINDI's success. Figure 3 provides visual evidence on the $16 \rightarrow 64$ transition. Without next-scale supervision, our INR renderer is still capable of generating predictions at the next finer scale due to its continous signal representation, but the predicted result provide no addition guidance for the next scale sampling process, leading the final generation with a much worse starting point. With the loss enabled, the next scale prediction better preserves the butterfly silhouette and wing structure, guiding the fine-scale trajectory toward a more natural image. The FID results confirm this trend quantitatively: removing the next-scale loss worsens FID from 2.19 to 2.75 for $32 \rightarrow 64$ and from 3.96 to 4.35 for $16 \rightarrow 64$.

Recipe ablations. Additional studies of per-scale noise, training allocation, step allocation, and INR modulation widths are reported in Appendix B. They support our selected per scale noise coefficients, training ratio, optimal step allocation, and hierarchical branch widths, respectively.

5.3 Matched comparison on ImageNet 256

We next scale CINDI to ImageNet 256×256 and compare with JiT at matched model scale and training epochs.

Table 2: ImageNet 256×256 CINDI vs JiT comparison using Section 5.1 protocol. Speedups are relative to the corresponding JiT model scale. Best performing numbers in each metric are bolded.

Variant	Scale ladder	Params	FID-50k ↓	FP32 img/s ↑	FP32 speedup	BF16 img/s ↑	BF16 speedup
JiT-L/16 (Li & He, 2026)	256	459M	2.79	1.25	1.00×	5.6002	1.00×
CINDI-L/16	128→256	461M	2.77	**1.77**	1.41×	**11.2379**	2.01×
JiT-H/16 (Li & He, 2026)	256	953M	2.29	0.62	1.00×	3.2145	1.00×
CINDI-H/16	128→256	956M	**2.28**	1.04	**1.67×**	7.0719	**2.20×**

As shown in Table 2, the best two-scale design transfers consistently to ImageNet 256×256. CINDI-L/16 reaches FID 2.77 versus 2.79 for JiT-L/16 while improving throughput by $1.41\times$ in FP32 and $2.01\times$ in BF16. CINDI-H/16 similarly reaches 2.28 versus 2.29 for JiT-H/16, with $1.67\times$ and $2.20\times$ speedups. As at 64×64, the small FID differences indicate comparable quality, whereas the throughput gains are substantial and consistent across precisions. CINDI also retains the expected

Table 3: ImageNet 256 × 256 comparison with modern generators. Published results follow JiT Tables 6–7 (Li & He, 2026); throughput values are our 1xH200 measurements, specifics in Appendix C. Best FID and throughput are bolded.

Method	Pretraining: token	perc.	self-sup.	Params	Epochs	FID ↓	img/s ↑
Latent diffusion							
DiT-XL/2 (Peebles & Xie, 2023)	SD-VAE	VGG	–	675+49M	–	2.27	2.1527
SiT-XL/2 (Ma et al., 2024)	SD-VAE	VGG	–	675+49M	–	2.06	–
REPA, SiT-XL/2 (Yu et al., 2025)	SD-VAE	VGG	DINOv2	675+49M	800	1.42	–
LightningDiT-XL/2 (Yao et al., 2025)	VA-VAE	VGG	DINOv2	675+49M	800	1.35	–
DDT-XL/2 (Wang et al., 2026)	SD-VAE	VGG	DINOv2	675+49M	400	1.26	–
RAE, DiT^{DH}-XL (Zheng et al., 2026)	RAE	VGG	DINOv2	839+415M	800 (+14)	**1.13**	–
Pixel non-diffusion							
JetFormer (Tschannen et al., 2025)	–	–	–	2.8B	500	6.64	–
FractalMAR-H (Li et al., 2025)	–	–	–	848M	600	6.15	0.9314
Pixel diffusion							
ADM-G (Dhariwal & Nichol, 2021)	–	–	–	554M	–	4.59	–
RIN (Jabri et al., 2023)	–	–	–	410M	–	3.42	–
SiD, UViT/2 (Hoogeboom et al., 2023)	–	–	–	2B	–	2.44	–
VDM++, UViT/2 (Kingma & Gao, 2023)	–	–	–	2B	–	2.12	–
SiD2, UViT/2 (Hoogeboom et al., 2025)	–	–	–	N/A	–	1.72	–
PixelFlow, XL/4 (Euler) (Chen et al., 2025)	–	–	–	677M	–	2.51	0.3703
PixelFlow, XL/4 (Dopri5) (Chen et al., 2025)	–	–	–	677M	–	1.98	0.0909
JiT-L/16 (Li & He, 2026)	–	–	–	459M	200	2.79	5.6002
JiT-H/16 (Li & He, 2026)	–	–	–	953M	200	2.29	3.2145
CINDI-L/16	–	–	–	461M	200	2.77	**11.2379**
CINDI-H/16	–	–	–	956M	200	2.28	7.0719

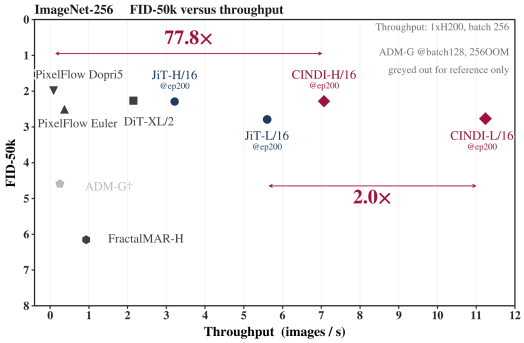

Figure 4: FID-50k vs inference throughput on ImageNet 256, Lower FID (upward) and higher throughput (rightward) are better. Arrows highlight the 2.0x throughput gain of CINDI-L/16 over JiT-L/16 and the 77.8x gap between CINDI-H/16 & PixelFlow Dopri5.

Transformer scaling trend: increasing model capacity from L/16 to H/16 improves FID from 2.77 to 2.28, closely paralleling JiT's improvement from 2.79 to 2.29.

5.4 Comparison with modern generators on ImageNet 256

Quality–throughput comparison. We also compare CINDI's 256 variants with other modern generative models. As shown in Table 3, with 200 training epochs and no tokenizer, perceptual network, or self-supervised pretraining, CINDI-L/16 reaches FID 2.77 at 11.2379 images/s, while CINDI-H/16 reaches 2.28 at 7.0719 images/s. By comparison, FractalMAR-H obtains 6.15 at 0.9314 images/s, and DiT-XL/2 nearly matches CINDI-H/16 in FID (2.27 versus 2.28) but runs at only 2.1527 images/s. PixelFlow (Chen et al., 2025) provides the closest progressive pixel-space comparison: its Euler configuration reaches FID 2.51 at 0.3703 images/s, so CINDI-H/16 improves quality with approximately $19\times$ higher throughput; its Dopri5 and stage-wise-CFG configuration improves FID to 1.98 but lowers throughput to 0.0909 images/s, making CINDI-H/16 $77.8\times$ faster at a 0.30 FID difference. When jointly considering quality and throughput among generative models, Figure 4 provides a direct visual comparison. Our CINDI variants occupy the upper-right region of Figure 4. CINDI-L/16 is the rightmost operating point, achieving the highest throughput among the compared models, while CINDI-H/16 further pushes toward higher generation quality while sustaining substantially higher throughput than methods of comparable quality.

Generation beyond the training resolution. Unlike most modern generative models, whose output grid is fixed to the training resolution, CINDI naturally supports generation beyond its training resolution through its coordinate-based implicit renderer. A CINDI-H/16 model trained at 256×256 can therefore be queried directly at 512×512 without fine-tuning. Appendix D and Figure 6 compare 512×512 direct render with nearest-neighbor upsampling of the native 256×256 output, showing smoother ear contours and more continuous whiskers.

6 Conclusion

We presented CINDI, a unified pixel-space diffusion model that treats spatial resolution as a progressive variable rather than a fixed property of the denoising trajectory. Across ImageNet 64×64 and 256×256, CINDI matches or edges out the FID of strong JiT baselines while delivering up to a $3\times$ increase in inference throughput under matched size, training, and sampling budgets. Among modern generative models, CINDI also occupies a competitive region that well balances quality and efficiency. Overall, CINDI provides a novel way to reduce generation cost through learned implicit bridging, without relying on latent compression or separately trained cascades, opening a new direction for efficient pixel-space generation.

AI USE STATEMENT

In this work, we used generative AI tools for grammar checking and sentence-level revision. They were not used to develop the research ideas, methodology, or scientific claims. The authors reviewed and edited all AI-assisted text and take full responsibility for the final content of this work.

REFERENCES

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. In *International Conference on Learning Representations*, 2023.

Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. PixelFlow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025.

Yinbo Chen, Oliver Wang, Richard Zhang, Eli Shechtman, Xiaolong Wang, and Michael Gharbi. Image neural field diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8007–8017, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Josh Susskind, and Navdeep Jaitly. Matryoshka diffusion models. *arXiv preprint arXiv:2310.15111*, 2023. doi: 10.48550/arXiv.2310.15111. URL https://arxiv.org/abs/2310.15111.

Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4246–4253, 2020.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Jonathan Ho, Chitwan Saharia, William Chan, David J. Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, 2023.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion: 1.5 FID on ImageNet512 with pixel-space diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Allan Jabri, David J. Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. In *International Conference on Machine Learning*, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577, 2022.

Jinseok Kim and Tae-Kyun Kim. Arbitrary-scale image generation and upsampling using latent diffusion model and implicit neural decoder. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9202–9211, 2024.

Diederik P. Kingma and Ruiqi Gao. Understanding diffusion objectives as the ELBO with simple data augmentation. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *Advances in Neural Information Processing Systems*, 2024.

Tianhong Li, Qinyi Sun, Lijie Fan, and Kaiming He. Fractal generative models. *Transactions on Machine Learning Research*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, 2024.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, 2020.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 3942–3951, 2018.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, 2015.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, 2022.

Vincent Sitzmann, Julien N. P. Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, 2020.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. JetFormer: An autoregressive generative model of raw images and text. In *International Conference on Learning Representations*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Shuai Wang, Zhi Tian, Weilin Huang, and Limin Wang. DDT: Decoupled diffusion transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations*, 2025.

Boyang Zheng, Nanye Ma, Shengbang Tong, and Saining Xie. Diffusion transformers with representation autoencoders. In *International Conference on Learning Representations*, 2026.

A Experimental Configurations

Table 4 summarizes the configurations used for the headline CINDI variants on ImageNet experiments. Unless a row differs across columns, the same setting is used at all model scales.

Table 4: Configurations for the principal CINDI experiments. The reported quality results select the best EMA and classifier-free guidance scale from the listed candidates; ablations modify only the configuration named in their respective tables.

Configuration	CINDI-B/4	CINDI-L/16	CINDI-H/16
Architecture			
Target image size	64	256	256
Transformer depth	12	24	32
Hidden dimension	768	1024	1280
Attention heads	12	16	16
Patch size	4	16	16
Patch bottleneck dimension	128	128	256
In-context class tokens	32	32	32
In-context start block	4	8	10
Implicit renderer			
Coordinate-MLP hidden dimension	256	256	256
FiLM-modulated layers	4	4	4
Fourier frequencies	8	8	8
Modulation-head widths	$(256, 192, 128, 64)$	$(256, 192, 128, 32)$	$(256, 192, 128, 32)$
Training			
Epochs	200	200	200
Warm-up epochs	5	5	5
Effective batch size	1024	1024	1024
Learning rate	2×10^{-4}	2×10^{-4}	2×10^{-4}
EMA decays		$\{0.9996, 0.9998, 0.9999\}$	
Denominator clamp t_ϵ	0.05	0.05	0.05
Class-label dropout	0.1	0.1	0.1
Multiscale training			
Scale ladder $\mathcal{R}$	$32 \to 64$	$128 \to 256$	$128 \to 256$
Training allocation ρ_s	$(0.40, 0.60)$	$(0.40, 0.60)$	$(0.40, 0.60)$
Flow-time intervals $[a_s, b_s]$	$[0, 1], [0.4, 1]$	$[0, 1], [0.4, 1]$	$[0, 1], [0.4, 1]$
Noise coefficients α_s	$(1, 3)$	$(1, 3)$	$(1, 3)$
Next-scale loss weight λ_{next}	0.1	0.1	0.1
Scale embedding	enabled	enabled	enabled
Sampling			
ODE solver	Heun	Heun	Heun
ODE steps per scale	$(36, 14)$	$(36, 14)$	$(36, 14)$
Total ODE steps K_{total}	50	50	50
CFG scale sweep		$[1.0, 4.0]$ with step 0.1	
CFG interval	$[0.1, 1]$	$[0.1, 1]$	$[0.1, 1]$
Throughput evaluation			
Hardware		1x NVIDIA H200 GPU	
Generation batch size	256	256	256
Warm-up runs	3	3	3
Timed inference runs	5	5	5

B ImageNet-64 Recipe Ablations

All ablations in Tables 5 and 6 use CINDI-B/4 with the $32 \to 64$ ladder, 200 training epochs, and $\lambda_{\text{next}} = 0.1$. Each panel in Table 5 changes one component of the selected training recipe while holding the others fixed. The repeated bold FID 2.19 therefore denotes the same reference configuration rather than separate cumulative improvements.

Table 5: One-factor recipe ablations for CINDI-B/4 on ImageNet 64×64 with the $32 \to 64$ ladder. The noise magnitudes are $\sigma_s = \alpha_s(r_s/256)$; bold rows identify the selected configuration.

(a) Per-scale noise

$(\alpha_{32}, \alpha_{64})$	$(\sigma_{32}, \sigma_{64})$	FID-50k ↓
Equal noise magnitude across scales		
$(2, 1)$	$(0.25, 0.25)$	4.18
$(4, 2)$	$(0.50, 0.50)$	3.44
$(6, 3)$	$(0.75, 0.75)$	3.21
Fix $\alpha_{32} = 1$; sweep α_{64}		
$(1, 1)$	$(0.125, 0.25)$	3.81
$(1, 2)$	$(0.125, 0.50)$	2.57
$(1, 3)$	$(0.125, 0.75)$	**2.19**
$(1, 4)$	$(0.125, 1.00)$	2.20
$(1, 5)$	$(0.125, 1.25)$	2.60

(b) Minibatch allocation

(ρ_{32}, ρ_{64})	FID-50k ↓
$(0.30, 0.70)$	2.35
$(0.35, 0.65)$	2.26
$(0.40, 0.60)$	**2.19**
$(0.50, 0.50)$	2.25

(c) INR modulation widths

Branch widths	FID-50k ↓
$(256, 192, 128, 64)$	**2.19**
$(256, 256, 256, 256)$	2.35

Scale-dependent noise. Panel (a) first tests a shared absolute noise magnitude at both scales; its best setting reaches FID 3.21. Fixing the coarse-scale coefficient at $\alpha_{32} = 1$ and sweeping only α_{64} improves FID to 2.19 at $(1, 3)$, a 1.02-point gain over the best shared-magnitude setting. The $(1, 3)$ and $(1, 4)$ settings are effectively tied at 2.19 and 2.20, respectively, whereas increasing to $(1, 5)$ clearly degrades FID to 2.60. We select $(1, 3)$ as the smallest fine-scale coefficient attaining the best performance.

Training allocation. Panel (b) varies the fraction of each minibatch assigned to the two scales. The selected $(0.40, 0.60)$ allocation reaches FID 2.19, compared with 2.25 for equal allocation and 2.35 when only 30% is assigned to the coarse scale.

INR modulation widths. Panel (c) compares the selected tapered widths $(256, 192, 128, 64)$ against four uniform 256-wide branches. Tapering improves FID from 2.35 to 2.19 while also using narrower later modulation branches.

Sampling-step allocation. We next fix the total sampling budget at $K_{32} + K_{64} = 50$ and vary only its allocation between the two scale segments in Table 6. Both FID-10k and BF16 throughput are evaluated at CFG 2.0; throughput follows our one-H200, batch-256 protocol.

The selected $(36, 14)$ split gives the lowest FID, 4.77; its neighboring $(34, 16)$ and $(38, 12)$ splits obtain 5.17 and 5.48. Throughput increases monotonically as steps move from the 64-pixel segment to the cheaper 32-pixel segment, rising from 29.57 to 73.12 images/s. The selected split therefore prioritizes generation quality while retaining 54.62 images/s.

Table 6: Sampling-step allocation for CINDI-B/4 with the $32 \rightarrow 64$ ladder. All splits use 50 total steps and CFG 2.0. FID is measured over 10k images; throughput is measured separately in BF16 on one H200 with batch size 256. The selected split is bolded.

Step split (K_{32}, K_{64})	FID-10k ↓	Throughput (img/s) ↑
$(4, 46)$	25.52	29.57
$(6, 44)$	19.18	30.40
$(8, 42)$	15.44	31.29
$(10, 40)$	12.88	32.34
$(12, 38)$	10.93	33.28
$(14, 36)$	9.77	34.43
$(16, 34)$	9.06	35.59
$(18, 32)$	8.43	36.81
$(20, 30)$	7.93	38.15
$(22, 28)$	7.50	39.64
$(24, 26)$	7.06	41.26
$(26, 24)$	6.69	42.88
$(28, 22)$	6.45	44.79
$(30, 20)$	6.12	46.78
$(32, 18)$	5.69	48.98
$(34, 16)$	5.17	51.41
$\mathbf{(36, 14)}$	**4.77**	**54.62**
$(38, 12)$	5.48	57.00
$(40, 10)$	5.58	60.27
$(42, 8)$	5.76	63.93
$(44, 6)$	6.54	68.15
$(46, 4)$	7.37	73.12

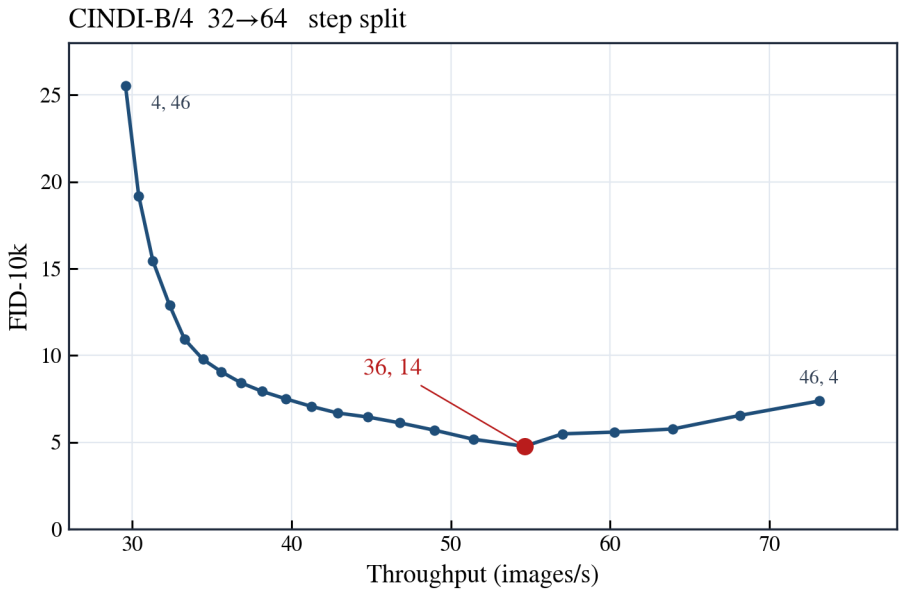

Figure 5: FID-10k versus BF16 throughput as the 50 sampling steps are redistributed between the 32- and 64-pixel segments. Labels denote (K_{32}, K_{64}); the selected $(36, 14)$ split is highlighted. Both axes use CFG 2.0 and the evaluation protocols reported in Table 6.

C Inference Speed Details

Table 7 reports the complete ImageNet 256×256 inference benchmark on one NVIDIA H200 GPU. Each throughput is the mean of five timed generation runs after three untimed warm-up runs; checkpoint loading is excluded. Speedups are normalized by the JiT-L/16 throughput at batch size 256; the final column additionally uses PixelFlow Dopri5 as the reference.

Table 7: Complete ImageNet 256×256 inference-speed benchmark. ADM-G uses the largest successful batch size; batch 256 exceeded 138 GB and did not complete.

Model	Batch	Precision	Sampling configuration	Throughput (img/s) ↑	Amortized per-image latency (s/image) ↓	Peak GB ↓	Speedup vs. JiT-L/16	Speedup vs. PixelFlow Dopri5
CINDI-L/16	256	BF16	Heun 36, 14, CFG	**11.2379**	**0.0890**	19.46	2.01×	123.63×
CINDI-H/16	256	BF16	Heun 36, 14, CFG	**7.0719**	**0.1414**	27.68	1.26×	77.80×
JiT-L/16	256	BF16	Heun 50, CFG	5.6002	0.1786	9.94	1.00×	61.61×
JiT-H/16	256	BF16	Heun 50, CFG	3.2145	0.3111	16.28	0.57×	35.36×
DiT-XL/2	256	BF16	250 DDPM steps, CFG	2.1527	0.4645	50.25	0.38×	23.68×
FractalMAR-Huge	256	FP16	Recursive 64, 16, 16, linear CFG	0.9314	1.0736	12.37	0.17×	10.25×
PixelFlow Euler	256	BF16	4 stages × 30 steps, global CFG	0.3703	2.701	58.66	0.07×	4.07×
PixelFlow Dopri5	256	BF16	Adaptive ODE, stage-wise CFG	0.0909	10.999	78.28	0.02×	1.00×
ADM-G	128	FP16; classifier FP32	250 steps, classifier scale	0.2485	4.0235	112.49	0.04×	2.73×

D Generation Beyond the Training Resolution

Figure 6 evaluates a CINDI-H/16 model trained only at 256×256. Directly querying its implicit renderer at 512×512 produces smoother boundaries around the ear and more continuous whiskers than nearest-neighbor upsampling of its native-resolution output. This comparison demonstrates that CINDI can render its learned continuous image function beyond the training grid without fine-tuning.

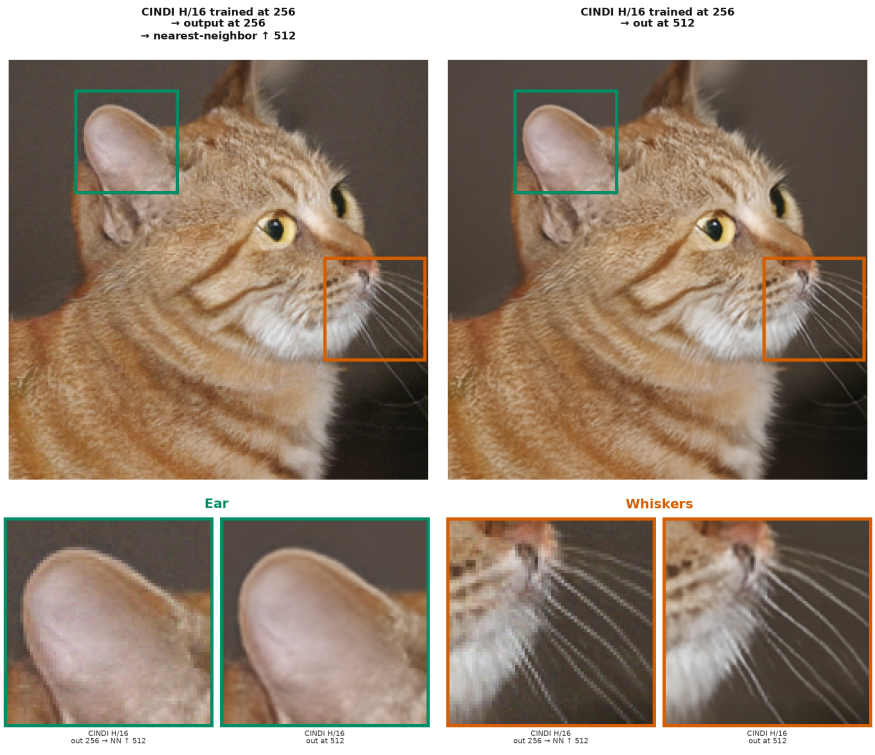

Figure 6: Generation beyond the training resolution. Left: CINDI-H/16 trained at 256×256, rendered at 256×256, then nearest-neighbor upsampled to 512×512. Right: the same model rendered directly at 512×512. The bottom row shows corresponding ear and whisker crops using shortened versions of the same labels.

E UNCURATED CLASS-CONDITIONAL SAMPLES



Figure 7: Uncurated 256×256 class-conditional ImageNet samples generated by CINDI-H/16. Samples within each selected class are shown without manual filtering or post-processing.

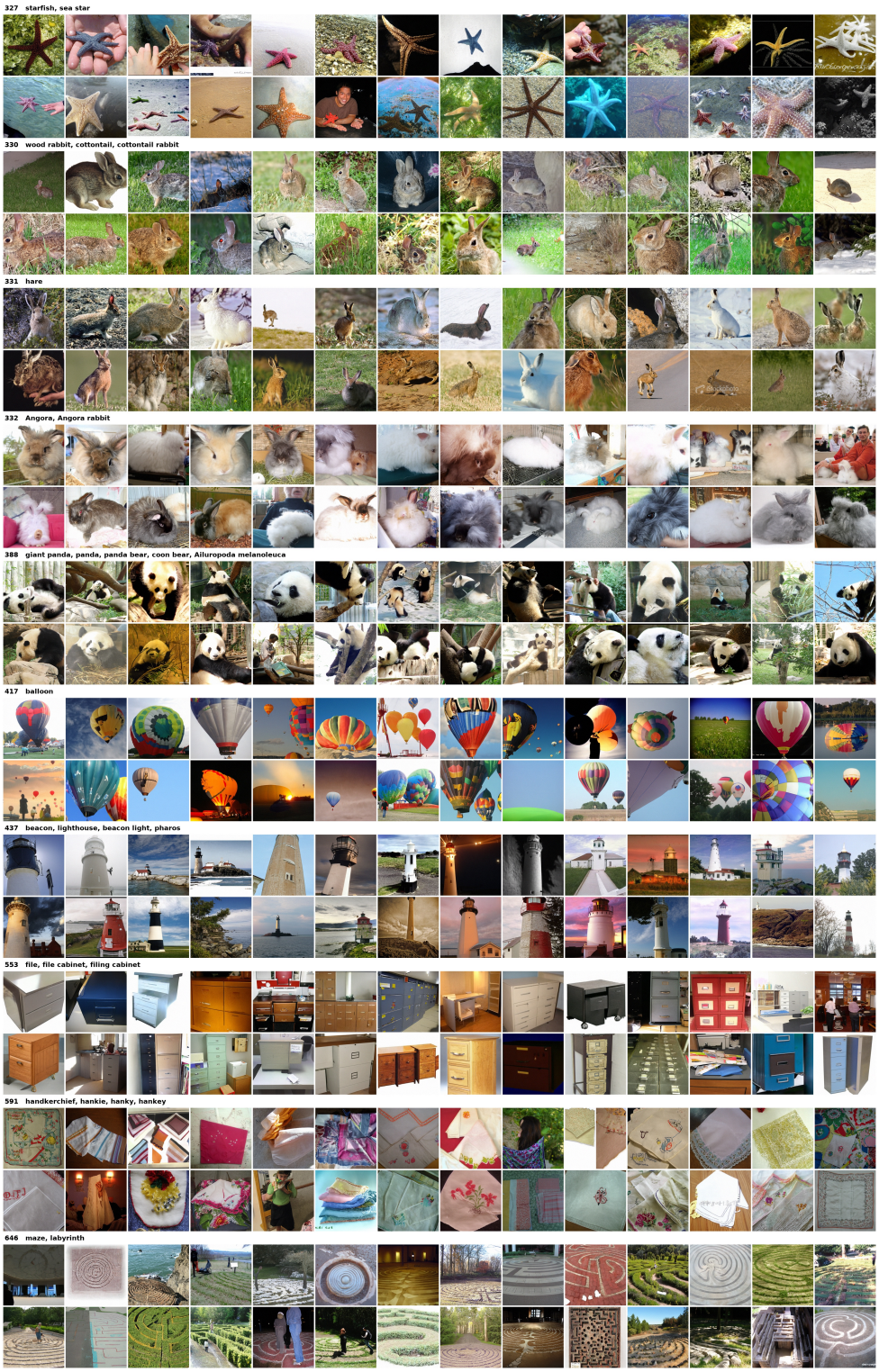

Figure 8: Additional uncurated 256×256 class-conditional ImageNet samples generated by CINDI-H/16.

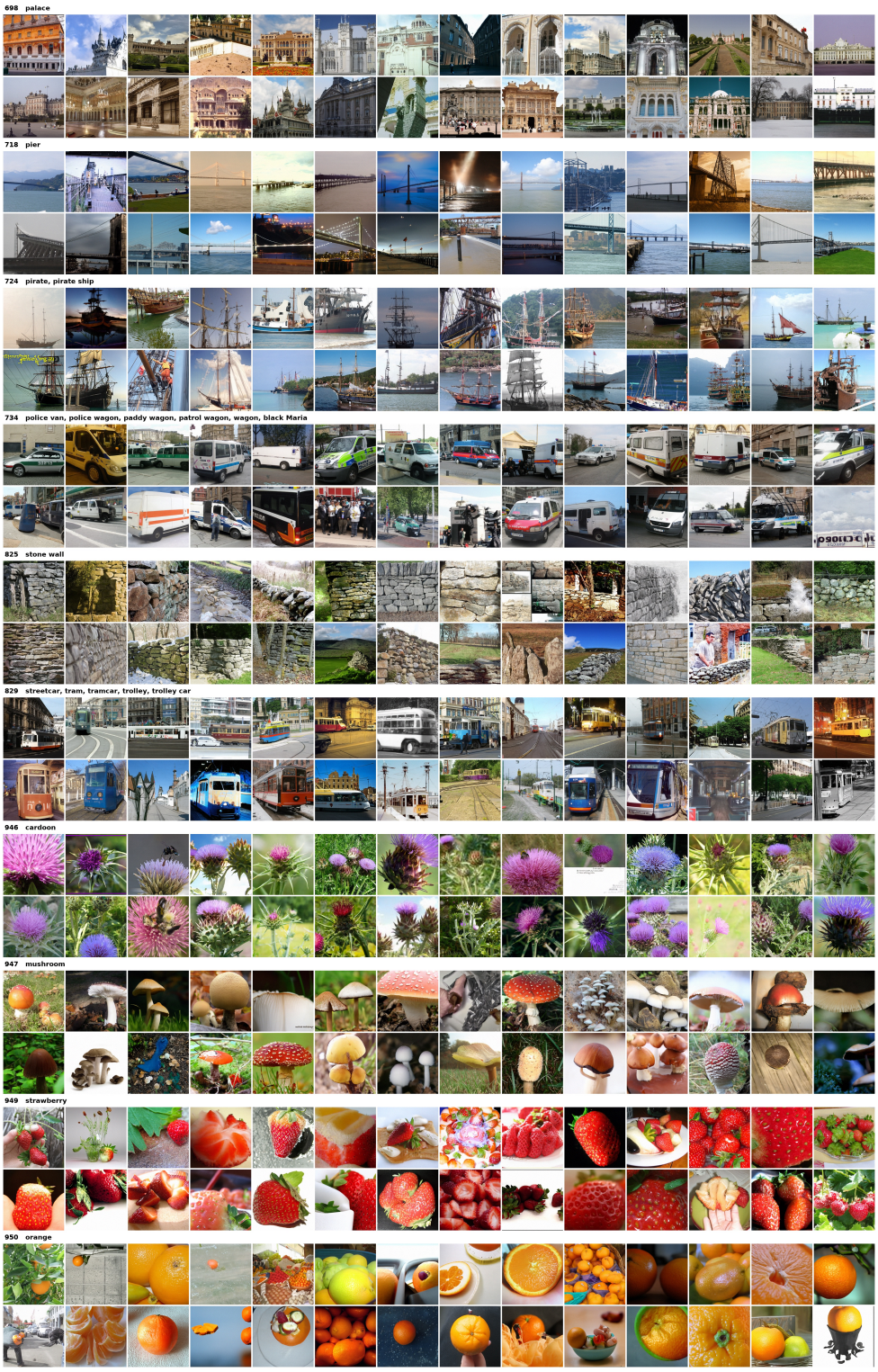

Figure 9: Additional uncurated 256×256 class-conditional ImageNet samples generated by CINDI-H/16.